INTEGRIS

Date 02/17/97

Number of pages including cover sheet 7

TO:

Prof. Silvio Micali

Phone 739 3039

Fax 617 ~~547~~ ~~3418~~

FROM: Chini Krishnan
Integris Security, Inc.
333 W. El Camino Real, Suite 270
Sunnyvale, CA 94086

Phone 408 738 4925
Fax Phone 408 738 2159

CC: Dr. Martin Hellman

REMARKS: ☐ Urgent ☐ For your review ☐ Reply ASAP ☐ Please Comment

Hi Silvio,

Trust you are doing well. Dr. Hellman suggested I send you the enclosed paper: a short but lucid description of our technology along with some product datasheets.

After we talked, we realized faxing would be quicker. Please do let me know if you don't receive all pages.

Best Regards,

Chini

Quick Introduction to Certificate Revocation Trees (CRTs)

This document assumes a basic understanding of secure hash functions, certificates, certificate revocation lists, etc.

Introduction

When a party in a cryptographic protocol (such as a web browser or server) verifies a certificate, it must perform a series of tests to determine whether it is acceptable. One critical test is to determine whether the certificate has been revoked.

The traditional solution to the revocation problem involves certificate revocation lists. Each CA publishes signed lists of revoked certificates. The verifier downloads these lists, checks the signature on the list, makes sure the list is recent, the date of the list, and searches the list to make sure that the serial number of the certificate in question is not on the list. CRLs cause a variety of problems, most notably that the verifier must have an up-to-date list of all revoked certificates from the CA, a list which could potentially become very large.

CRTs make it possible for the verifier (any Internet or other application using certificates) to obtain a compact, time-critical proof that any certificate has not been revoked. A system based on CRTs consists of three major components: CRT issuance, confirmation issuance (of certificate status) and verification (of such confirmation).

CRT Issuance

To produce a CRT one must first obtain a set of all revoked certificates from the CA(s) in the tree. These would typically come from CRLs, but revocations could also come from other sources such as users revoking their own certificates. Actual revocation policies are determined by each CA.

Any revoked certificate can be uniquely identified by its CA's public key (or the hash of the public key) and the certificate serial number. For any CA, there may be zero or more revoked certificates.

Imagine a tree with three CAs with public key hashes $CA_1 < CA_2 < CA_3$, where CA_1 has revoked 4 certificates (156, 343, and 344), CA_2 has revoked no certificates, and CA_3 has 1 revoked certificate (987). The CRT issuer can now make the following statements about certificate serial number X from a CA whose public key hash is CA_X:

If: $-\infty < CA_X < CA_1$	Then: Unknown CA (revocation status unknown)
If: $CA_X = CA_1$ and $-\infty \le X < 156$	Then: X is revoked if and only if $X = -\infty$.
If: $CA_X = CA_1$ and $156 \le X < 343$	Then: X is revoked if and only if $X = 156$.
If: $CA_X = CA_1$ and $343 \le X < 344$	Then: X is revoked if and only if $X = 343$.
If: $CA_X = CA_1$ and $344 \le X < \infty$	Then: X is revoked if and only if $X = 344$.
If: $CA_1 < CA_X < CA_2$	Then: Unknown CA (revocation status unknown).
If: $CA_X = CA_2$ and $-\infty \le X < \infty$	Then: X is revoked if and only if $X = -\infty$.

If: $CA_2 < CA_X < CA_3$	Then: Unknown CA (revocation status unknown).
If: $CA_X = CA_3$ and $-\infty \leq X < 987$	Then: X is revoked if and only if $X = -\infty$.
If: $CA_X = CA_3$ and $987 \leq X < \infty$	Then: X is revoked if and only if $X = 987$.
If: $CA_3 < CA_X < \infty$	Then: Unknown CA (revocation status unknown).

Note that for any CA_X and X there is a single appropriate statement above which provides all known information about whether X is revoked. Furthermore, no statement can accidentally be applied to an inappropriate certificate.

The CRT issuer now builds a set of data structures which express the statements above. Note that only two basic types of data structures are required: those which specify a range of unknown CAs and those which specify a range of certificates in which the lower endpoint is revoked and all larger certificates in the range are valid.

The CA then builds a binary hash tree with hashes of the data structures as leaves. For the example above, there would be eleven leaf nodes ($N_0..N_{0,10}$), where $N_{0,i}$ is the secure hash of the structure expressing the i+1th statement above.

$$\begin{tikzcd}
N_{0,0} \arrow[dr] & & & & \\
& N_{1,0} \arrow[rdd] & & & \\
N_{0,1} \arrow[ur] & & & & \\
& & N_{2,0} \arrow[rdddd] & & \\
N_{0,2} \arrow[dr] & & & & \\
& N_{1,1} \arrow[ruu] & & & \\
N_{0,3} \arrow[ur] & & & & \\
& & & N_{3,0} \arrow[rdddddddd] & \\
N_{0,4} \arrow[dr] & & & & \\
& N_{1,2} \arrow[rdd] & & & \\
N_{0,5} \arrow[ur] & & & & \\
& & N_{2,1} \arrow[ruuuu] & & \\
N_{0,6} \arrow[dr] & & & & \\
& N_{1,3} \arrow[ruu] & & & \\
N_{0,7} \arrow[ur] & & & & \\
& & & & N_{4,0} \\
N_{0,8} \arrow[dr] & & & & \\
& N_{1,4} \arrow[rdd] & & & \\
N_{0,9} \arrow[ur] & & & & \\
& & N_{2,2} \arrow[r] & N_{3,1} \arrow[ruuuu] & \\
N_{0,10} \arrow[r] & N_{1,5} \arrow[ru] & & &
\end{tikzcd}$$

Wherever two arrows converge on a single node (such as $N_{2,0}$ is produced from $N_{1,2}$ and $N_{1,3}$), the right-hand node is computed as the secure hash of the left-hand two nodes with the upper node hashed first. For example, $N_{2,0}$ would equal $H(N_{1,2} | N_{1,3})$, where "|" denotes concatenation. A nodes with a single arrow is equal to the node to its left (i.e., $N_{3,1} = N_{2,2}$). Note that the root node ($N_{4,0}$) is a function of all leaf nodes ($N_{0,0} \ldots N_{0,10}$). After producing the entire tree, the issuer digitally signs the root node along with supporting information (like the date and time of the tree issuance).

Confirmation Issuance and Verification

To determine the revocation status of a certificate, the verifier needs the appropriate statement and proof that the statement is correct. The role of a confirmation issuers is to provide a verifier these two components.

Expressing the statement is easy; that's just the appropriate leaf data structure. To assure the statement's validity, the verifier needs cryptographic proof that the leaf representative of the revocation status of the certificate is a the leaf in a properly signed tree. This assurance consists of the signed root node and a set of intermediate nodes which cryptographically bind the appropriate leaf node to the root node.

For example, for certificate 600 from CA_1, the appropriate statement is:

If: $CA_X = CA_1$ and $344 \leq X < \infty$ Then: X is revoked if and only if $X = 344$.

The verifier can hash this statement structure to get $N_{0,5}$. The supporting nodes in this example are $N_{0,4}$, $N_{1,3}$, $N_{2,0}$, and $N_{3,1}$. The verifier can now use the secure hash function H to compute:

$$N_{1,2} = H(N_{0,4} | N_{0,5})$$
$$N_{2,1} = H(N_{1,2} | N_{1,3})$$
$$N_{3,0} = H(N_{2,0} | N_{2,1})$$
$$N_{4,0} = H(N_{3,0} | N_{3,1})$$

Finally, the verifier can check that $N_{4,0}$ was properly signed by the trusted tree issuer with the correct date, etc. If the original statement structure was tampered with or if any of the intermediate hashes were changed, the verifier will get a different value for $N_{4,0}$ than was $N_{4,0}$ signed by the tree issuer.

Advantages of a CRT based system

A CRT based approach has many advantages, notably:

- Verification does not require knowledge of entire CRLs.
- CRL caching and other optimizations are not required.

- Certificate holders can efficiently confirmations of their own certificates so that verifiers do not need to make any additional network connections.
- The system works with existing certificates, CRLs, etc.
- The size of the supporting hashes is approximately $20\log_2(N)$, where N is the number of revoked certificates. The system works efficiently even if N is in the billions or trillions.
- One tree with one signed root can provide proofs for all certificates in a chain, so only one public key operation can verify multiple certificates.
- The system supports CA revocation.
- All supporting information found in CRLs can be included including the revocation date/time, reason for revocation, etc.

DATA SHEET

Certium Server™

The Certium Server is an OEM application intended for integration with Certificate Servers that provides the ability to efficiently validate digital certificates to Internet applications. The server implements the tree-issuance and confirmation issuance portions of a certificate validity management system based on Certificate Revocation Trees.

The server includes C source code, developers' documentation and consulting assistance from Integris to complete the integration work.

BENEFITS

- Minimal network and processing overhead
- Scales to support billions of digital certificates
- Fully exportable
- Leading browser support planned

Highlights

- Comprehensive Intranet confirmation issuer for any Intranet Certificate Server. Primary features include:
  - a) Issuance of Certificate Revocation Tree based on both local certificate server or the globally operated Integris service.
  - b) Issuance of certificate confirmation to requesting applications.
  - c) Forms-based administration interface for easy addition to HTTP based Certificate Servers.
- An OEM product, source code available for tighter integration and customization.
- Implementation based on open standards: PKCS-7, RSA, SHA-1, X-509, ASN.1, HTTP.
- Single and multi-threaded implementations included.

DATA SHEET

Certium Application Toolkit™

The Certium application toolkit is a developer's toolkit that provides end user applications such as Web browsers and electronic mail clients with the ability to check the validity of digital certificates in real-time. The toolkit implements the verifier component of a certificate validity management system based on Certificate Revocation Trees. A typical verifier consists of functionality for the creation of certificate confirmation requests, the validation of certificate confirmation response, and the location of the nearest confirmation issuer.

The toolkit includes C source code, developers' documentation and consulting assistance from Integris to complete the integration work.

CERTIUM BENEFITS

- Certificate Validation transparent to end users
- Scales to support billions of digital certificates
- Fully exportable
- Leading browser support planned

Highlights

- Designed to work with both the Integris operated Certium Service and any Certificate Server or Certificate Management System that incorporates the Certium Server.
- Non-repudiable certificate status proof may be obtained by certificate holder, recipient or third-party.
- A transparent solution, end user typically unaware of validation.
- Thread-safe implementation, available on both Windows and UNIX.
- Cryptographic code bases supported include Tipem/BSAFE and SSLEay, CryptoAPI 2.0 planned for 4Q 1996 release.
- Implementation based on open standards: PKCS-7, RSA, SHA-1, X-509, ASN.1, HTTP.
- Toolkit license includes both development and redistribution rights.